# RNNs for Visual Question Answering

**Shashank Solomon**
University of California San Diego
s2solomon@eng.ucsd.edu
A53281006

**Srinithya Nagiri**
University of California San Diego
snagiri@eng.ucsd.edu
A53277642

**Gayathri Cuddappa Sudhinder**
University of California San Diego
gcuddappa@eng.ucsd.edu
A53266365

**Chandhini Grandhi**
University of California San Diego
cgrandhi@eng.ucsd.edu
A53272378

## Abstract

This project takes inspiration from the paper "Show, Ask, Attend and Answer: A strong baseline for Visual Question and Answering"[1], which provides a new baseline for visual question and answering task. The task of this project is to create a model which when given an image and natural language question regarding the image as input, predicts the top five most probable natural language answers to the question. To achieve this, we trained a convolutional neural network based on Resnet architecture, LSTM and a classifier to produce the final answers. With this method, we were able to achieve an accuracy of **61%** which closely resembles to the accuracy described in the paper.

## 1 Introduction

Visual question and answering (VQA) is an interesting multi-discipline research challenge combining computer vision, natural language processing (NLP) and deep learning. We need NLP for understanding the question and generate the answers. The main characteristic in VQA is that search and reasoning is performed over the content of the image. That is, to answer if humans are present in the image, the model must be able to detect objects. To answer if it is raining, the model needs to classify the scene. Almost all the above tasks has been addressed in the field of Computer vision. VQA is fascinating because it requires the model to understand the image it sees.

One research topic which is closely related to VQA is image caption generation, which generates captions when given image as input. The disadvantage of such a model is that the system may not understand the image completely as it is easy to generate a single caption that holds for a large collection of images. This problem is less severe in VQA as it is possible to ask narrow questions which forces the model to narrow down and understand the image.

In this project, we train a residual network [3] to detect features from the image and a long-short term memory units (LSTM) [2] to encode the question. To get multiple features based on the state of LSTM we used a soft attention mechanism [4]. A final classifier then takes the multiple features and the final state of LSTM to produce the most probable answers. A high- level architecture of the model is provided in Figure 1. We used accuracy as our evaluation metric and our model achieves an accuracy of **61%** which is same as the accuracy obtained in the reference paper [1].

# 2 Description

We explain our approach more in detail in this section. The visual question answering system is treated as a classification problem in terms of probability. Given an image $I$ and a question $q$ in the form of natural language we need to estimate the most likely answer â based on the context of the image. This can be written mathematically as:

$$\hat{a} = \arg_a \max P(a \mid I, q)$$

where a $\in$ {a1,a2,...an}. The answers are chosen to be the most frequent from the training set.

## 2.1 Architecture

Figure 1 shows the overall high level architecture of the model.

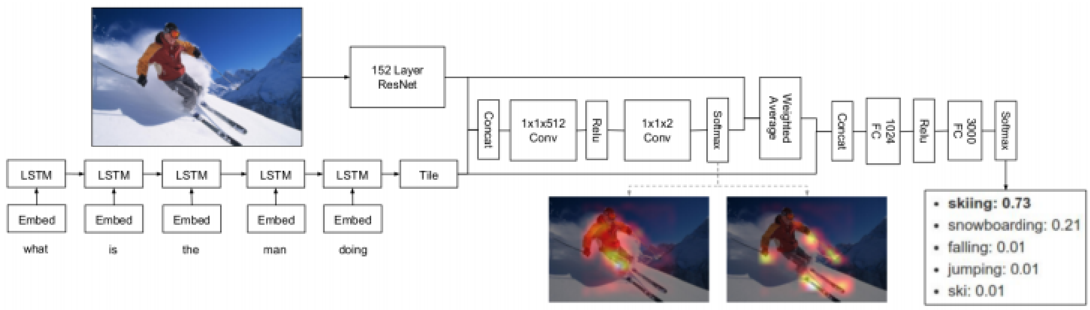


Figure 1: Overall architecture of the model.

### 2.1.1 Overview of the model presented

We use a convolutional neural network based on ResNet[3] to embed the image. The input question is tokenized and embedded and fed to a multi-layer LSTM. The concatenated image features and the final state of LSTMs are then used to compute multiple attention distributions over image features. The concatenated image feature glimpses and the state of the LSTM is fed to two fully connected layers to produce probabilities over answer classes.

The following steps are followed for the VQA task and individual model architecture is explained in depth.

### 2.1.2 Image embedding

We used a convolutional neural network based on ResNet[5] to embed the image. The figure 2 depicts the ResNet model. It contains 8 layers with weights- the first five are convolutional and the remaining three are fully connected. The output of the last fully-connected layer is fed to a 1000-way softmax which produces a distribution over the 1000 class labels. Our network maximizes the multinomial logistic regression objective, which is equivalent to maximizing the average across training cases of the log-probability of the correct label under the prediction distribution.

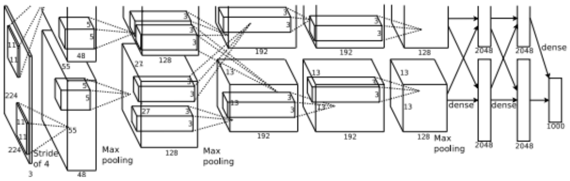


Figure 2: Architecture of CNN

We trained a Convolutional neural network model based on residual network archicture [5] to compute a high level representation $\phi$ of the input image $I$

$$\phi = \text{CNN(I)}$$

where $\phi$ us a three dimensional tensor from the last layer dimensional tensor from the last layer of the residual network[3] before the final pooling layer with 14 * 14 * 2048 dimensions. We furthermore perform l2 normalization on the depth (last) dimension of image features which enhances learning dynamics.

### 2.2 Question Embedding

We tokenize and encode a given question q into word embeddings

$$E_q = \{\text{e1, e2, ....eP}\}$$

where ei $\in$ R$^D$, D is the length of the distributed word representation, and P is the number of words in the question.

**LSTM:** The LSTM, one of the recurrent neural networks. Compared with an usual recurrent network, the LSTM can capture autoregressive structures of arbitrary lengths.

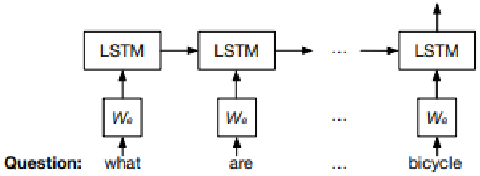


Figure 3: LSTM based question model

Figure 3: LSTM based question model

The figure 3 shows the architecture of LSTM based question model [2]. The essential structure of LSTM unit in a memory cell $c_t$ which reserves the state of the sequence. At each step, the LSTM unit takes one input vector $x_t$ and updates the memory cell $c_t$, then output a hidden state $h_t$. The update process uses the gate mechanism. A forget fate $f_t$ controls how much information from past state $c_t - 1$ is preserved. An input gate $i_t$ controls how much the current input $x_t$ updates the memory cell. An output gate $o_t$ controls how much information of the memory is fed to the output as hidden state. The detailed update process is as follows:

$$\begin{gathered}
i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
f_t = \sigma(W_{xi}f_t + W_{hf}h_{t-1} + b_f) \\
o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
c_t = f_t c_t{-}_1 + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
h_t = o_t \tanh(c_t)
\end{gathered}$$

where i,f,o,c are input gate, forget gate, output gate and memory cell respectively. The weight matrix and bias are parameters of the LSTM and are learned on the training data. Given the question q = [$q_1$,...$q_T$], where $q_t$ is one hot vector representation of word at position $t$, we first embed the words to a vector space through an embedding matrix $x_t = W_e q_t$. Then for every time step, we feed the embedding vector of words in the question to LSTM:

$$\begin{gathered}
x_t = W_e q_{t,}\ t \in 1,2,3..T \\
h_t = \text{LSTM}(x_t)_,\ t \in 1,2,3...T
\end{gathered}$$

As shown in figure 3 , the question "what are sitting in the basket on a bicycle" is fed into the LSTM. Then the final hidden layer is taken as the representation vector for the question.

As mentioned above, similarly in our model the embeddings are then fed to a long short term memory

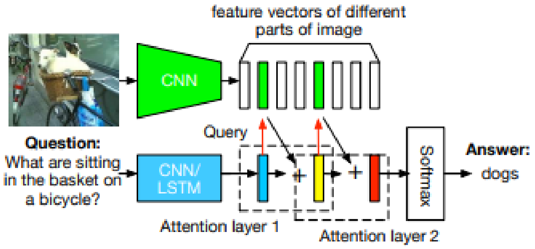
$$s = \text{LSTM(Eq)}$$

we use the final state of the LSTM to represent the equation.

### 2.2.1 Stacked Attention Network (SAN)

The SAN consists of three major components:

1. The image model, which uses a CNN to extract high level image representations, e.g. one vector for each region of the image
2. The question model, which uses a CNN or a LSTM to extract a semantic vector of the question
3. The stacked attention model, which locates, via multi-step reasoning, the image regions that are relevant to the question for answer prediction.


feature vectors of different parts of image
CNN
Query
Question:
What are sitting in the basket on a bicycle?
CNN/ LSTM
Attention layer 1
Attention layer 2
Softmax
Answer:
dogs

(a) Stacked Attention Network for Image QA

Figure 4: Stacked Attention network for Image QA

The figure 4 shows the overall architecture of the SAN. It first uses the question vector to query the image vectors in the first visual attention layer, then combine the question vector and the retrieved image vectors to form a refined query vector to query the image vectors again in the second attention layer. The higher-level attention layer gives a sharper attention distribution focusing on the regions that are more relevant to the answer. Finally, we combine the image features from the highest attention layer with the last query vector to predict the answer.

Similar to Stacked Attention network model architecture, we compute multiple attention distributions over the spatial dimensions of the image features.

$$\alpha_{ct} \propto \exp \text{F}_c \, (\text{s},\phi_t) \ni \sum_{l-1}^{L} \alpha_{ct} = 1$$
$$\text{x}_c = \sum \alpha_{ct} \; \phi_t$$

Each image feature glimpse x$_c$ is the weighted average of the image features $\phi$ over all the spatial locations l = 1,2....L. The attention weights $\alpha_{ct}$ are normalized separately for each glimpse c = 1,2..C

In practice F = [F$_1$, F$_2$...F$_C$] is modeled with two layers of convolution. Consequently F$_i$ share parameters in the first layer. We solely rely on different initialization to produce diverse attention distributions.

### 2.2.2 Classifier

Finally we contantenate the image glimpses along with the LSTM state and apply nonlinearities to produce most probable answer classes.

$$\text{P(a}_i\text{| I,q)} \propto \exp \text{G}_i \; \text{(x,s)}$$

where

$$\text{x} = [\text{x}_1, \text{x}_2, .....,\text{x}_c]$$

G = [G$_1$,G$_2$ ...G$_M$] in practice is modeled with two fully connected layers. Our final loss is defined as follows.

$$L = \frac{1}{K} \sum_{k=1}^{K} - \log P(a_k | I,q)$$

Note that we average the log-likelihoods over all the correct answers $a_1$,$a_2$,...$a_k$.

# 3 Experimental Setting

## 3.1 Dataset

**VQA 1.0** We evaluate our model on both balanced and unbalanced versions of VQA dataset. VQA 1.0 [6] is consisted of 204,721 images form the MS COCO dataset [7]. We evaluate our models on the real open ended challenge which consists of 614,163 questions and 6,141,630 answers. The dataset comes with predefined train, validation, and test splits. There is also a 25% subset of the the test set which is referred to as test-dev split. For our experiments we used the train set as training data and reported the results on the validation set.

**Evaluation metric** We evaluate our models on the open ended task of VQA challenge with the provided accuracy metric.

$$Acc(a) = \frac{1}{K} \sum_{k=1}^{K} \min\left(\left(\sum_{1 \le j \le K,\, j \ne k} 1(a = a_j)/3\right) 1\right)$$

where $a_1$,$a_2$,...$a_K$ are the correct answers provided by the user and K=10. Intuitively, we consider an answer correct if at least three annotators agree on the answer. To get some level of robustness we compute the accuracy over all 10 choose 9 subsets of ground truth answers and average.

## 3.2 Training Parameters

Total number of trainable parameters is 19882502. We run training for 50 epochs, batch size 128, adam optimizer with initial learning rate 0.0001.

## 3.3 Implementation details

We train our model on the MS COCO v1 (2014) training images and open-ended questions dataset which has 82,783 images. We evaluate accuracy and loss on the MS COCO v1 (2014) validation dataset which has 40,504 images. A vocabulary is built using all the training questions and all the questions are tokenized using this vocabulary. We use resnet-152 pretrained model to extract feature maps of input images. Each feature map is of size 14x14. Input images are rescaled to 448x448 before going through ResNet. The input questions are tokenized and fed to an LSTM network. The results of both ResNet and LSTM networks are concatenated and sent through additional convolutional layers with non-linearities. We are able to achieve an accuracy of 61.1% on the validation dataset. We have used pytorch to write the model.

## 3.4 Hardware used

We trained our model on UCSD Data Science and Machine Learning Platform. For most part of our project, we used the Nvidia's GeForce GTX 1080 Ti GPU along with 11172MB GRAM.

# 4 Results

## 4.1 Classifier accuracy

The graph shown in figure 5 is the variation of accuracy and loss as the iterations increases. It can be seen that our model does fairly well, the accuracy increases as the number of iteration increases and it replicates the accuracy obtained in the reference paper.

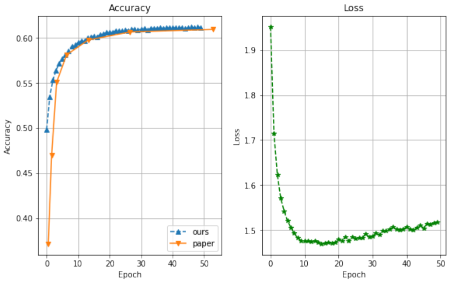
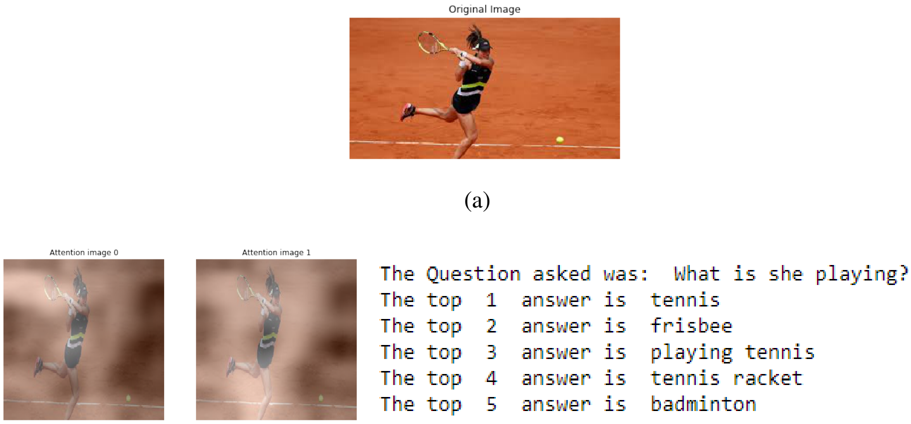


Figure 5: Accuracy and loss

## 4.2 Predicted Samples

Several samples of the predicted answers are available in the figures below. The input images are not part of the test set as well. We downloaded images from the internet and tested the performance of the model on that. In figures 6 to 8, (a) shows the actual image as input which is fed to the network, (b) shows the multiple attention distributions obtained from the over the concatenated image features and the final state of LSTM and (c) shows the five most probable answers for the question asked based on the input image.



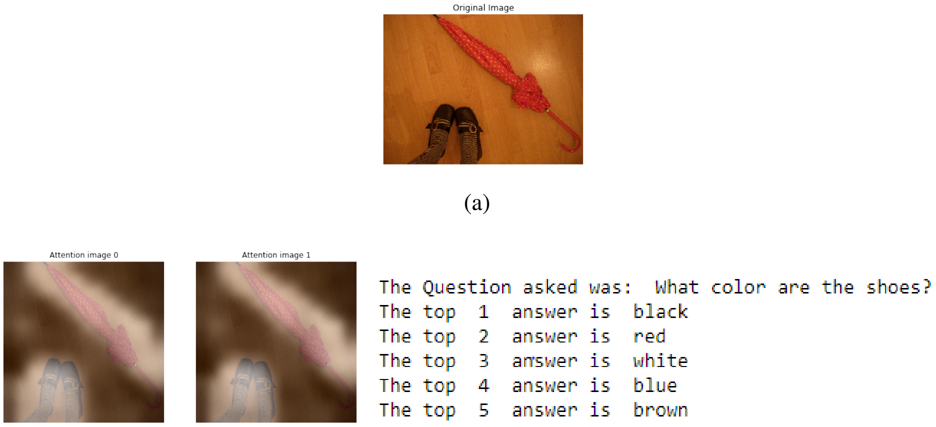
```
The Question asked was:  What is she playing?
The top  1  answer is  tennis
The top  2  answer is  frisbee
The top  3  answer is  playing tennis
The top  4  answer is  tennis racket
The top  5  answer is  badminton
```

(a)

(b)

(c)

Figure 6: Tennis image, attention model and QA

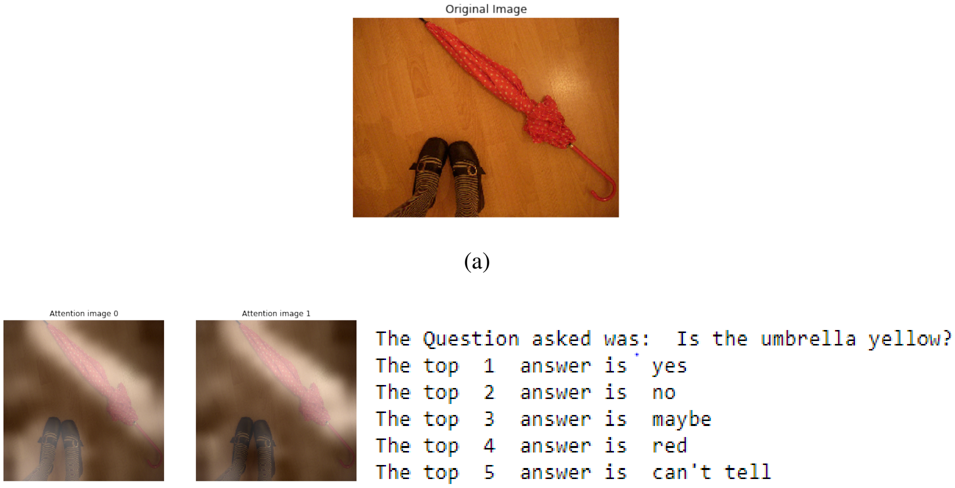


```
The Question asked was:  What color are the shoes?
The top  1  answer is  black
The top  2  answer is  red
The top  3  answer is  white
The top  4  answer is  blue
The top  5  answer is  brown
```

(a)

(b) (c)

Figure 7: Shoes image, attention model and QA



```
The Question asked was:  Is the umbrella yellow?
The top  1  answer is  yes
The top  2  answer is  no
The top  3  answer is  maybe
The top  4  answer is  red
The top  5  answer is  can't tell
```

(a)

(b) (c)

Figure 8: Umbrella image, attention model and QA

As it can be seen, the model gets most of the answers correct. But like any other deep learning model, it makes mistakes which can be seen in figure 8.

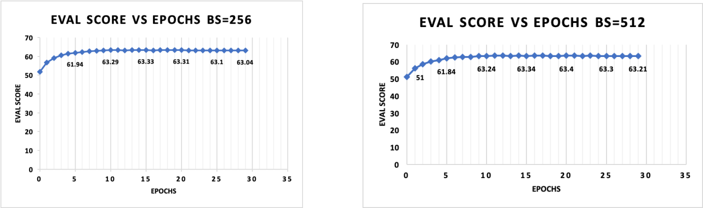
### 4.3 Model comparison

As mentioned in the discussion section, we tried different implementations before settling down on implementing the reference paper [1]. Table 1 shows the comparison of the two different version of Show-attend and Tell[4]

We then implemented the Bottom-up-attention[9] and figure 9 shows the accuracy obtained for two different batch sizes - 256 and 512.

Table 1: Accuracy of Show, attend and tell[4]

| Metrics | without Beam Search | with Beam search |
|---|---|---|
| Bleu1 | 0.64 | 0.698 |
| Bleu2 | 0.456 | 0.528 |
| Bleu3 | 0.278 | 0.39 |
| Bleu4 | 0.210 | 0.288 |
| Meteor | 0.201 | 0.232 |
| Rouge | 0.483 | 0.514 |
| cider | 0.58 | 0.824 |



(a) (b)

Figure 9: Accuracy of bottom up attention model[9]

Based on all this knowledge and failed implementations, as seen in Figure 5 we knew our model had done fairly well.

## 5 Discussion

As the results show we achieved fairly good results. We finalized on implementing the above model after a number of attempts on training networks based on existing work. To be specific, we implemented two versions of Show attend and tell, with and without Beam Search and procedded to implement Stacked Attention network and then Bottom-up-attention- VQA. Below are some of the details of the models we implemented:

**Show-attend and tell, with Beam Search** We referred to this repository which implements the python3 version of Show,attend and tell: Neural image caption generation with Visual attention [4]. This was one of the reference papers given to us as a part of Project A. The input to this model is an image, and the output is a sentence describing the content of the image. It uses a convolutional neural network to extract visual features from the image, and uses a LSTM recurrent neural network to decode these features into a sentence. A soft attention mechanism is incorporated to improve the quality of the caption. This project is implemented using the Tensorflow library, and allows end-to-end training of both CNN and RNN parts.

We used the pre-trained model presented in the paper, evaluated the results on the validation MS COCO dataset. We spent significant amount of time to implement this and fixed all the background issues which include package dependencies, dataset pruning etc. We realized this does not deal with VQA and just generates the captions. We achieved the accuracy present in the paper and proceeded to fine tune the parameters. The output and other generated files are placed in our GitHub repository here.

**Show, attend and tell, without Beam Search** We implemented the Show, attend and tell [4] which takes inspiration from this github repository. This implements the same paper but the architecture of the model is different. This model uses pretrained VGG with 19 layers to extract image features of size 14x14. The features are then passed to an LSTM network with 16 layers to get 16 word captions. This implements a soft attention scheme while generating each successive word in the caption. We tried multiple values for batch size(64,128,etc.), hidden units(512,1024) and learning rates(0.001,0.0005,etc.) but were unable to get it to the reported meteor score in the paper which is

23.90. We were able to achieve a meteor score of around 20. This is just a caption generator though and can not accomplish VQA directly. So we decided to try other models. The results obtained are placed in our GitHub repository here.

**Stacked Attention Network-VQA** We implemented the Stacked Attention Network [8] which is available in this repository. We trained an LSTM and CNN based questions models and they use two attention layers. The repository does not contain any pre-trained model and we had to train it. We trained the model for 6 hours, and obtained a model which ran completely for 10000 iterations. After it got completed, we finally launched another pod for 12 hours, and obtained a model which ran for 25000 iterations.We almost had to wait until the training was done. Using this model,we set out to evaluate it on the validation data set.

During the evaluation set, we learn that the code is faulty, and there had been an open git issue regarding the same. This can also be seen in the óutput.txt́file in the sanvqatensorflow folder.The intermediate json output files generated did not have predictions for all question ids in annotation file. Hence, the evaluation step could not go through. We tried debugging about the same, and saw that there is a difference of around 30000 samples having this flaw. We realized this is not going to work because of the huge difference and dropped this reference. The work we did and the output generated are placed in our GitHub repository here.

**Bottom-up-attention-VQA** We implemented the VQA using bottom-up-attention model [9] which takes inspiration from this repository.
In the original implementation of Bottom-Up Attention VQA, there is an image captioning model and a VQA model which takes as input a possibly variably-sized set of k image features, such that each image feature encodes a salient region of the image. Output from the faster R-CNN bottom-up attention model is the image with various bounding boxes. Each bounding box is labeled with an attribute class followed by an object class. However, in captioning and VQA only the feature vectors are utilised not the predicted labels. But in this implementation, few changes were made to decrease the training time like using only a fixed number of objects per image (K=36). And, ReLU activation function is used as the non-linear activation unit instead of gated tanh.
The training took around 4 hours for 30 epochs. Initially we did training with the default parameters. We achieved an evaluation score of 63.21 for validation set for 30 epochs which is nearly equal to evaluation score mentioned in the repo. Then we modified the batch size from 512 to 256 and performed training. We achieved an evaluation score of 63.04 for validation set.The guidelines in Project A mentiones about using an architecture which uses RNN and LSTM. The bottom-up attention VQA uses CNN and we wanted to try and understand RNN as it was new to us, so we decided to stop implementing this model. The ouput and results obtained are placed in our GitHub repository here.

### 5.1 Conclusion

Our work presented above trains a CNN based on ResNet architecture, LSTM, a Stacked attention network and a classifier for the task of Visual Question and Answering. Both good predictions and bad predictions occurred during training model. The ultimate performance of trained model on the test set is around 61%.

### 5.2 Future work

In our future work, we would like to use club vectors instead of the current word embeddings for the question embedding task as they encode more information. We could also add weight normalization to enhance accuracy . We could also use Adamax optimizer, as it is more suitable for sparsely updated parameters (e.g. embeddings) and makes the algorithm robust to noise in the gradients. We could also explore the task of training an independent network to learn from the image captions that were generated [4] to model for the VQA task.

## 6 References

[1] Kazemi, V. & Elqursh, A. (2017) Show,Ask,Attend and Answer: A strong baseline for Visual Question Answering. *arXiv preprint arXiv: 1704.03162* , 2017

[2] Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Computation*, 9: 1735-1780,1997.

[3] He, K. & Zhang, X. & Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770-778, 2016.

[4] Xu, K. & Ba, J. & Kiros, R. & Cho , K. & Courville A. C. & Salakhutdinov R. & Zemel R. S. & Begnio Y. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*,2015.

[5] Krizhevsky, A. & Sutskever, I. & Hinton G. E. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[6] Antol, S. & Agrawal, A. & Lu, J. & Mitchell, M. & Batra D. & Zitnick, L. C. & Parikh D. Vqa : Visual question answering. In *International Journal of Computer Vision*, 2015.

[7] Lin Y. T. & Maire M. & Belongie J. S. & Hays J. & Persona, P. & Ramanan, D. & Dollar, P. & Zitnick, L. C. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[8] Yang, Z. & He, X. & Gao, J. & Deng, L. & Smola, A. Stacked attention networks for image question answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern recognition*, 2016.

[9] Anderson, P. & He, X. & Buehler, C. & Teney, D. & Johnson, M. & Gould, S. & Zhang, L. Bottom-up and top-down attention for image captioning and vqa. In proceedings of *IEEE Conference Computer Vision and Pattern Recognition*, 2018